- ページ表の大きさはページ・サイズに反比例する．したがって，ページ表に要するメモリ（およびその他の資源）は，ページを大きくすればするほど節約できる．
- 大きなページを2次記憶との間でやりとりすること（ネットワークを介する場合もある）は，小さなページをやりとりするよりも効率が良い．

（ページ・サイズが大きいと，キャッシュ・アドレスのアドレス変換においても有利である：8.8節参照）

ページ・サイズを小さくしようとする場合，その主な動機は記憶領域の節約である．ページ・サイズを小さくすれば，記憶領域の無駄を少なくできる．特に，仮想記憶上の連続領域がページ・サイズの倍数に等しくならないときはそうである．この場合，ページ内に生ずる未使用メモリのことを**内部フラグメンテーション** (internal fragmentation) と呼ぶ．プロセスごとに3つのセグメント（テキスト，ヒープ，およびスタック）があるとして，プロセスごとの記憶領域の無駄は，平均してページ・サイズの約1.5倍ほどある．これは，主記憶を何Mバイトも持っていて，かつページ・サイズが2~8Kバイト程度のマシンであれば，見過ごすことのできる量である．しかし，ページ・サイズが32Kバイトを越えるほど大きくなれば，主記憶はもちろん，2次記憶の領域も大量に浪費されることになる．そしてそれは，入出力のバンド幅をも浪費する．最後に触れておかなければならないことは，プロセスの立ち上げ時間のことである．すなわち，小さなプロセスがたくさんある場合，ページ・サイズが非常に大きいと，それだけプロセスの起動に時間がかかり，全体の性能を落とすことになる．

アドレス変換の高速化

ページ表は通常かなりの大きさになるので，主記憶上に割り付けられ，それ自身ページングの対象になったりもする．これは，メモリ・アクセスのすべてに2倍の手間が必要になることを意味している．すなわち，まず物理アドレスを知るために1回目のメモリ・アクセスを行い，実際にデータを得るために2回目のメモリ・アクセスを行う必要があるということである．これではなんとも高価に過ぎる．

1つの解決法は，前回のアドレス変換の内容を記憶しておくことである．こうすることで，次もまた同じページ内のアドレスに参照する場合には，アドレス・マッピングの過程を省略できる．ここでは，これよりもう少し一般的な解決法を述べるが，それはまたもや参照の局所性を活用した方法である．すなわち，参照に局所性がある場合は，アドレス変換にも局所性があるはずである．そこで，これらのアドレス変換を特別なキャッシュ中に保持しておけば，メモリ・アクセスを2回行う必要は格段に減少する．この特別なアドレス変換用のキャッシュを**アドレス変換バッファ** (translation-lookaside buffer) または**TLB**と呼ぶ．単に変換バッファ (translateion buffer) またはTBと呼ばれることもある．TLBの1エントリは，キャッシュのエントリと同様，タグ部とデータ部とから成る．タグ部には仮想アドレスが保持され，データ部には物理ペ

ージ・フレーム番号に加え，保護用フィールド，使用ビット，ダーティ・ビットが保持される．ページ表の1エントリに対して，その物理ページ・フレーム番号や保護用フィールドの書き換えを行う際は，そのエントリがTLB上にないことをOSが確かめる必要がある．TLB上にあることを知らずにページ表の方だけを書き換えてしまうと，システムは正しく動作しなくなる．ここで，ダーティ・ビットは，対応するページがダーティであることを意味するのであって，TLB中のエントリそのものがダーティであるとか，キャッシュのあるブロックがダーティであるとかということを意味するのではないことに注意していただきたい．図8.24は，典型的なTLBの諸元である．

ブロック・サイズ	4〜8バイト（1ページ表エントリ）
ヒット時間	1クロック・サイクル
ミス・ペナルティ	10〜30クロック・サイクル
ミス率	0.1%〜2%
TLBサイズ	32〜8192バイト

図8.24 TLBに関する記憶階層の諸元 TLBとは，ページ表を参照して仮想アドレスから物理アドレスへのアドレス変換を行う際のキャッシュとして働くものである．

キャッシュと仮想記憶とを組み合わせることには多少の困難がつきまとう．そこで，アーキテクチャ上での工夫が行われることになる．まず，仮想アドレスは，物理アドレスでキャッシュにアクセスする以前にTLBに送られなければならない．これは，キャッシュのヒット時間がアドレス変換に必要な時間だけ引き延ばされることを意味している（または，パイプラインのタイミングが引き延ばされる：第6章参照）．ヒット時間を減らす1つの方法は，ページ内オフセットでキャッシュをアクセスすることである．この部分は，アドレスの仮想部分の中で変換を要しない部分である．キャッシュのアドレス・タグを読み出している間に，アドレスの仮想部分（ページフレーム・アドレス）をTLBに送り，変換する．その後，TLBから出力された物理アドレスとキャッシュ・タグとの間でアドレス比較を行う．TLBはキャッシュ・アドレス・タグよりも小容量で高速のメモリであるので，TLBの読み出しを同時進行させれば，キャッシュのヒット時間を引き延ばすことにはならない．ただ，この方式の欠点は，ダイレクト・マップ方式のキャッシュをページ・サイズよりも大きくできない，という点である．これとは別の方法である仮想アドレス・キャッシュについては，8.8節で述べる．

8.6 記憶保護および仮想記憶システムの例

多重プログラミングの発明によって，プログラム間でのメモリの共有と保護の必要性が新たに生まれた．これらは，今日のコンピュータにおける仮想記憶の技術と深く結びついている．そこで本節では，仮想記憶システムの例を2つ